MN

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/892,633	06/28/2001	Randal F. Templeton	219.40067X00 (ATSK)	4474

7590 03/05/2007

Kenyon & Kenyon
1500 K Street, N.W.
Suite 700
Washington, DC 20005-1257

EXAMINER	
TRAN, QUOC A	
ART UNIT	**PAPER NUMBER**
2176	

SHORTENED STATUTORY PERIOD OF RESPONSE	MAIL DATE	DELIVERY MODE
3 MONTHS	03/05/2007	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

If NO period for reply is specified above, the maximum statutory period will apply and will expire 6 MONTHS from the mailing date of this communication.

Office Action Summary	**Application No.** 09/892,633	**Applicant(s)** TEMPLETON ET AL.	
	Examiner Tran A. Quoc	**Art Unit** 2176	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *08 December 2006*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-18* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-18* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO/SB/08)
Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413)
Paper No(s)/Mail Date. _____ .
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

DETAILED ACTION

1) This is a **Non-final** rejection in response to Remarks filed on 12-08-2006.

2) Claims 1-18 remain in the application. Claims 1, 7, 10, 13 and 16 are independent claims.

3) Effective filing date is 6-28-2001.

Response to Argument

4). Applicant's arguments, in the Remarks filed 12-08-2006 with respect to claims 1-18 have been considered but are moot in view of the new ground(s) of rejection. This office action is a Non-Final Rejection in order to give the applicant sufficient opportunity to response to the new line of rejection.

Claim Rejections - 35 USC § 103

5) The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> *(a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.*

5-1) Claims 1-18 are rejected under 35 U.S.C. 103(a) as being unpatentable over of Chen et al US006507856B1 filed 01-05-1999 (hereinafter Chen), in view of Uhler et al. US US007089560B1 filed 07-24-2000 (hereinafter Uhler).

Regarding independent claim 1, Chen teaches **a console engine to receive requests for web pages and messages to be send to web pages.** Specifically, Chen discloses XML parser and DTD parser for receiving and returning a message from a browser (Chen col. 1, lines 35-50, also Col. 3, line 65 through col. 4, line 10, and Fig. 7 item 305 and 315).

Using the broadest reasonable interpretation, the Examiner reads the claimed **a console engine** as equivalent to XML parser as taught by Chen, and because Applicant's invention specification discloses "A console engine is used to parse a incoming XML data element" (see Applicant's invention the Abstract).

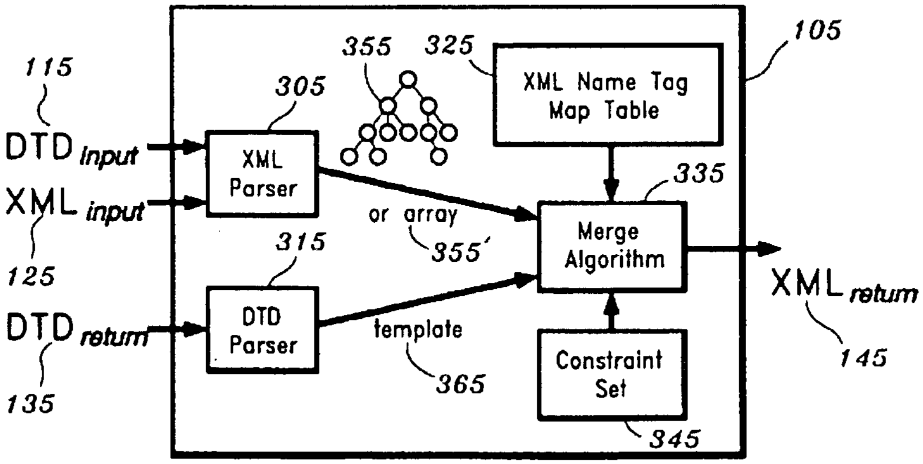
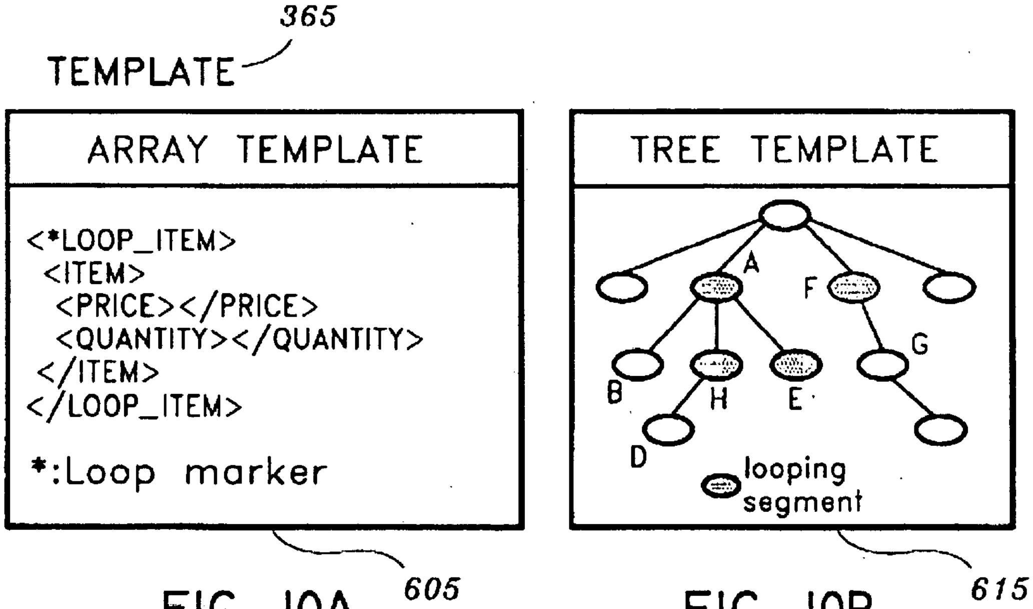


FIG. 7

In addition, Chen teaches **an XML repository connected to the console engine having a plurality of parts of web pages, and the console engine is to extract a template for a web page from one of said requests** Specifically, Chen discloses a standard XML parser item 305, may be a client side application, which may serialize tree elements into an array of hyper-text markup language (HTML) components 355', or a server side stand-alone application, which takes the input XML 125 and DTD 115, and generates an intermediate structure, a tree 355 or an array 355', which serves as part of the input data to a merge algorithm 335. which may construct the tree structure 355 (See FIGS. 10A and 10B). After parsing the return document DTD 135, the DTD parser 315 creates a template 365 in either array format 605 or tree structure 615, as shown in FIGS. 10A and 10B, respectively (Chen Col. 6, lines 5-20 also fig, 10A and 10B).



```
<*LOOP_ITEM>
 <ITEM>
  <PRICE></PRICE>
  <QUANTITY></QUANTITY>
 </ITEM>
</LOOP_ITEM>
*:Loop marker
```

FIG. 10A

FIG. 10B

Using the broadest reasonable interpretation, the Examiner reads the claimed **a plurality of parts** as equivalent to serialize tree elements as taught by Chen.

In addition, Chen does not expressly teach, but Uhler teaches **said console engine is to retrieve at least one application handler.** For example, Uhler discloses Application Programming Interface (API) called a handler using a delegation based object model (Uhler col. 6, lines 1-10).

In addition, Chen does not expressly teach, but Uhler teaches **a plurality of HTM/XML templates, said retrieved application handler being registered to said extracted template and said application handler to modify said template ant to generate a part of said requested web page and incorporate that part into the template to form the web page.** Specifically, Uhler discloses the filter handler uses a set of HTML/XML templates to process content and the final filter performing the XML to HTML conversion, which is consistent with the ultimate consumer of the content, and deliver to the requestor (Uhler col. 14, line 60 through col. 15, line 10). Also, Uhler discloses Application Programming Interface (API) called a handler using a delegation based object model. The handlers that provide application functionality are resolved and loaded at run time. Mechanisms are provided for composing application modules, encouraging code reuse and design. Information specific to an entire application is gathered in one place, and made available to all of the handlers, simplifying server modification and configuration (Uhler col. 6, lines 1-10).

It would have been obvious to a person of ordinary skill in the art at the time the invention was made to have modified Chen's document exchanging and merging system, includes a plurality of HTM/XML templates, said retrieved application handler being registered to said extracted template and said application handler to modify said template ant to generate a part of said requested web page and incorporate that part into the template to form the web page as taught by Uhler. One of the ordinary skill in the art would have been motivated to modify this combination, because Chen and Uhler are from the same field of endeavor of providing an architecture for creating extensible and scalable web applications, and provides a server object handler uses a set of HTML/XML templates to process content and the final filter performing the XML to HTML conversion, which is consistent with the ultimate consumer of the content, and deliver to the requestor (Uhler col. 14, line 60 through col. 15, line 10).

Regarding independent claims 7 and 10, the rejection of claim 1 is fully incorporated. In addition, Chen teaches **combining the plurality of parts for the web page with the template to form the web page; and transmitting the web page to the web browser for display.** Specifically, Chen discloses a merging algorithm, which is implemented to merge the message with the return template for providing a return message to the browser having portions of the return template with data entered therein. (Chen Col. 1, lines 45-50).

In addition, Chen teaches **accessing an XML repository for a template for the web page.** Specifically, Chen discloses a standard XML parser item 305, may be a client side application, which may serialize tree elements into an array of hyper-text markup language (HTML) components 355', or a server side stand-alone application, which takes the input XML 125 and DTD 115, and generates an intermediate structure, a tree 355 or an array 355', which serves as part of the input data to a merge algorithm 335, which may construct the tree structure 355 (See FIGS. 10A and 10B). After parsing the return document DTD 135, the DTD parser 315 creates a template 365 in either array format 605 or tree structure 615, as shown in FIGS. 10A and 10B, respectively (Chen Col. 6, lines 5-20 also fig, 10A and 10B).

In addition, Chen does not expressly teach, but Uhler teaches **at least one application handler that is required to modify the template.** Specifically, Uhler discloses Application Programming Interface (API) called a handler using a delegation based object model (Uhler col. 6, lines 1-10). Also, Uhler discloses the filter handler uses a set of HTML/XML templates to process content and the final filter performing the XML to HTML conversion, which is consistent with the ultimate consumer of the content, and deliver to the requestor (Uhler col. 14, line 60 through col. 15, line 10).

It would have been obvious to a person of ordinary skill in the art at the time the invention was made to have modified Chen's document exchanging and merging system, includes at least one application handler that is required to modify the template as taught by Uhler. One of the ordinary skill in the art would have been motivated to modify this combination, because Chen and Uhler are from the same field of endeavor of providing an architecture for creating extensible and scalable web applications, and provides a server object handler uses a set of HTML/XML templates to process content and the final filter performing the XML to HTML conversion, which is consistent with the ultimate consumer of the content, and deliver to the requestor (Uhler col. 14, line 60 through col. 15, line 10).

Regarding independent claims 13 and 16, Chen teaches **receiving an incoming XML data element from a source web page, parsing the incoming XML data element based on delimiters to determine the source web page.** Specifically, Chen discloses a first parser for receiving a message from a browser (Chen col. 1, lines 35-50). Also, Chen discloses a standard XML parser 305 takes the input XML 125 and DTD 115, and generates an intermediate structure, a tree 355 or an array 355', which serves as part of the input data to a merge algorithm 335. The XML parser 305 may be a client side application, which may serialize tree elements into an array of hypertext markup language (HTML) components 355', or a server side stand-alone application, which may construct the tree structure 355 (See FIGS. 10A and 10B). After parsing the return document DTD 135, the DTD parser 315 creates a template 365 in either array format 605 or tree structure 615, as shown in FIGS. 10A and 10B, respectively (Chen col. 6, lines 5-20 also Fig. 7). Also, Chen discloses a merging algorithm, which is implemented to merge the message with the return template for providing a return message to the browser having portions of the return template with data entered therein. (Chen Col. 1, lines 45-50).

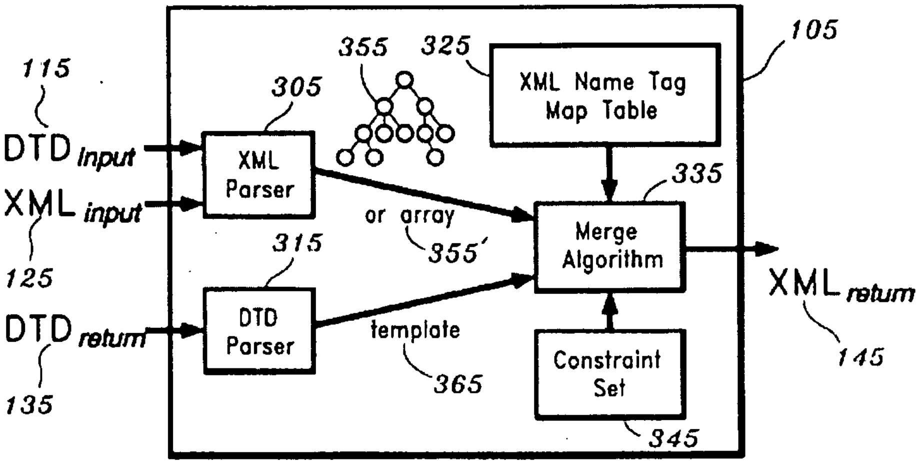


FIG. 7

In addition Chen teaches **creating a pretoken from the data in the incoming XML data element; concatenating the pretoken to a token to form a modified XML data clement.** Specifically, Chen discloses a standard XML parser 305 takes the input XML 125 and DTD 115, and generates an intermediate structure, a tree 355 or an array 355', which serves as part of the input data to a merge algorithm 335. The XML parser 305 may be a client side application, which may serialize tree elements into an array of hypertext markup language (HTML) components 355', or a server side stand-alone application, which may construct the tree structure 355 (See FIGS. 10A and 10B). After parsing the return document DTD 135, the DTD parser 315 creates a template 365 in either array format 605 or tree structure 615, as shown in FIGS. 10A and 10B, respectively (Chen col. 6, lines 5-20 also Fig. 7). Also, Chen discloses a merging algorithm, which is implemented to merge the message with the return template for providing a return message to the browser having portions of the return template with data entered therein. (Chen Col. 1, lines 45-50).

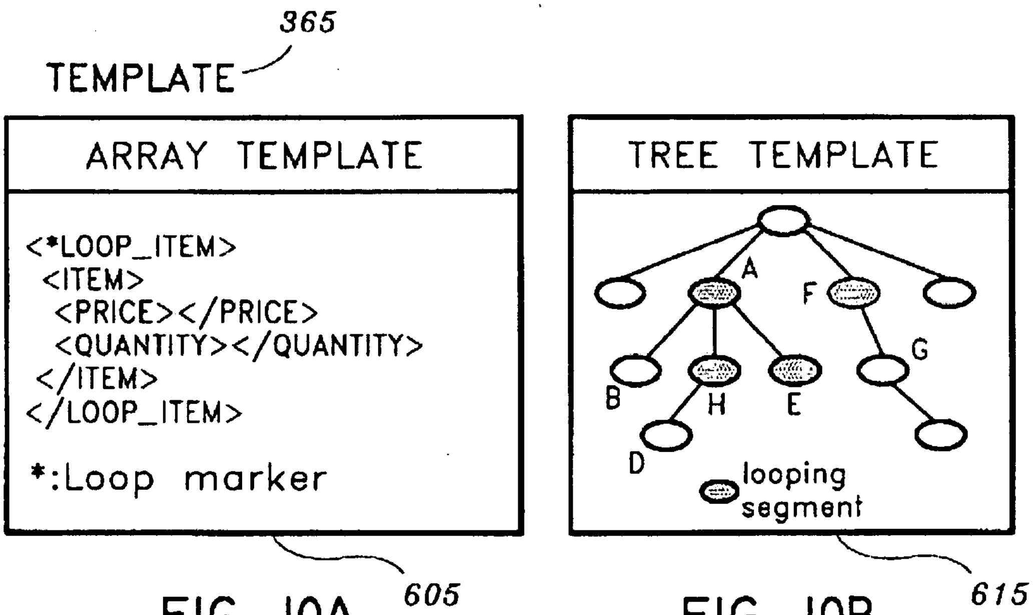


FIG. 10A

FIG. 10B

In addition, Chen does not expressly teach, but Uhler teaches **a destination web page, and data to be received by the destination web page, and said modified XML data clement including a template for the destination web page**. For example, Uhler discloses the Request object 104 contains all of the information that pertains to client's URL request as well as methods that encapsulate the HTTP protocol (Uhler col. 7, lines 20-35). Also Uhler discloses the filter handler uses a set of HTML/XML templates to process content and the final filter performing the XML to HTML conversion, which is consistent with the ultimate consumer of the content, and deliver to the requestor (Uhler col. 14, line 60 through col. 15, line 10). Also, Uhler discloses Application Programming Interface (API) called a handler using a delegation based object model. The handlers that provide application functionality are resolved and loaded at run time. Mechanisms are provided for composing application modules, encouraging code reuse and design. Information specific to an entire application is gathered in one place, and made available to all of the handlers, simplifying server modification and configuration (Uhler col. 6, lines 1-10).

It would have been obvious to a person of ordinary skill in the art at the time the invention was made to have modified Chen's parsing the incoming XML data element based on delimiters to determine the source web page, to include data to be received by the destination web, to include a destination web page, and data to be received by the destination web page, and said modified XML data clement including a template for the destination web page as taught by Uhler. One of the ordinary skill in the art would have been motivated to modify this combination, because Chen and Uhler are from the same field of endeavor of providing an architecture for creating extensible and scalable web applications, and provides a server object handler uses a set of HTML/XML templates to process content and the final filter performing the XML to HTML conversion, which is consistent with the ultimate consumer of the content, and deliver to the requestor (Uhler col. 14, line 60 through col. 15, line 10).

Regarding claim 2, Chen teaches **a web browser to request the web page from the console engine and display the web page**. Specifically, Chen discloses a first parser for receiving a message from a browser (Chen col. 1, lines 35-50). Also, Chen discloses a merging algorithm, which is implemented to merge the message with the return template for providing a return message to the browser having portions of the return template with data entered therein. (Chen Col. 1, lines 45-50).

Regarding claims 3-4, Chen does not expressly teach, but Uhler teaches **an XML repository to contain the plurality of parts of web pages, the plurality of HTML/XML templates and a plurality of said application handlers, and a console API to transmit the web page to a web browser.** For example, Uhler discloses the Request object 104 contains all of the information that pertains to client's URL request as well as methods that encapsulate the HTTP protocol (Uhler col. 7, lines 20-35). Also Uhler discloses the filter handler uses a set of HTML/XML templates to process content and the final filter performing the XML to HTML conversion, which is consistent with the ultimate consumer of the content, and deliver to the requestor (Uhler col. 14, line 60 through col. 15, line 10). Also, Uhler discloses Application Programming Interface (API) called a handler using a delegation based object model. The handlers that provide application functionality are resolved and loaded at run time. Mechanisms are provided for composing application modules, encouraging code reuse and design. Information specific to an entire application is gathered in one place, and made available to all of the handlers, simplifying server modification and configuration (Uhler col. 6, lines 1-10).

It would have been obvious to a person of ordinary skill in the art at the time the invention was made to have modified Chen's parsing the incoming XML data element based on delimiters to determine the source web page, to include data to be received by the destination web, to include an XML repository to contain the plurality of parts of web pages, the plurality of HTML/XML templates and a plurality of said application handlers as taught by Uhler. One of the ordinary skill in the art would have been motivated to modify this combination, because Chen and Uhler are from the same field of endeavor of providing an architecture for creating extensible and scalable web applications, and provides a server object handler uses a set of HTML/XML templates to process content and the final filter performing the XML to HTML conversion, which is consistent with the ultimate consumer of the content, and deliver to the requestor (Uhler col. 14, line 60 through col. 15, line 10).

Regarding claim 5, Chen teaches **console engine parses said message to identify delimiters contained in the message, the source web page, and data contained in the message.** Specifically, Chen discloses a first parser for receiving a message from a browser (Chen col. 1, lines 35-50). Also, Chen discloses a standard XML parser 305 takes the input XML 125 and DTD 115, and generates an intermediate structure, a tree 355 or an array 355', which serves as part of the input data to a merge algorithm 335. The XML parser 305 may be a client side application, which may serialize tree elements into an array of hypertext markup language (HTML) components 355', or a server side stand-alone application, which may construct the tree structure 355 (See FIGS. 10A and 10B). After parsing the return document DTD 135, the DTD parser 315 creates a template 365 in either array format 605 or tree structure 615, as shown in FIGS. 10A and 10B, respectively (Chen col. 6, lines 5-20 also Fig. 7). Also, Chen discloses a merging algorithm, which is implemented to merge the message with the return template for providing a return message to the browser having portions of the return template with data entered therein. (Chen Col. 1, lines 45-50).

In addition, Chen does not expressly teach, but Uhler teaches **a destination web page**. For example, Uhler discloses the Request object 104 contains all of the information that pertains to client's URL request as well as methods that encapsulate the HTTP protocol (Uhler col. 7, lines 20-35).

It would have been obvious to a person of ordinary skill in the art at the time the invention was made to have modified Chen's parsing the incoming XML data element based on delimiters to determine the source web page, to include data to be received by the destination web, to include a destination web page as taught by Uhler. One of the ordinary skill in the art would have been motivated to modify this combination, because Chen and Uhler are from the same field of endeavor of providing an architecture for creating extensible and scalable web applications, and provides a server object handler uses a set of HTML/XML templates to process content and the final filter performing the XML to HTML conversion, which is consistent with the ultimate consumer of the content, and deliver to the requestor (Uhler col. 14, line 60 through col. 15, line 10).

Regarding claim 6, Chen teaches **console engine concatenates the data from the message with the template to create a modified XML data element that is displayed the web browser**. Specifically, Chen discloses a first parser for receiving a message from a browser (Chen col. 1, lines 35-50). Also, Chen discloses a standard XML parser 305 takes the input XML 125 and DTD 115, and generates an intermediate structure, a tree 355 or an array 355', which serves as part of the input data to a merge algorithm 335. The XML parser 305 may be a client side application, which may serialize tree elements into an array of hypertext markup language (HTML) components 355', or a server side stand-alone application, which may construct the tree structure 355 (See FIGS. 10A and 10B). After parsing the return document DTD 135, the DTD parser 315 creates a template 365 in either array format 605 or tree structure 615, as shown in

FIGS. 10A and 10B, respectively (Chen col. 6, lines 5-20 also Fig. 7). Also, Chen discloses a merging algorithm, which is implemented to merge the message with the return template for providing a return message to the browser having portions of the return template with data entered therein. (Chen Col. 1, lines 45-50).

Regarding claims 8-9, and 11-12, the rejection of claim 4 is fully incorporated. In addition, Chen does not expressly teach, but Uhler teaches **converting the template after combining the plurality of parts for the web page with the template to form the web page into HTML so as to be displayed by the browser**. For example, Uhler discloses the filter handler uses a set of HTML/XML templates to process content and the final filter performing the XML to HTML conversion, which is consistent with the ultimate consumer of the content, and deliver to the requestor (Uhler col. 14, line 60 through col. 15, line 10). Also, Uhler discloses Application Programming Interface (API) called a handler using a delegation based object model. The handlers that provide application functionality are resolved and loaded at run time. Mechanisms are provided for composing application modules, encouraging code reuse and design. Information specific to an entire application is gathered in one place, and made available to all of the handlers, simplifying server modification and configuration (Uhler col. 6, lines 1-10).

It would have been obvious to a person of ordinary skill in the art at the time the invention was made to have modified Chen's parsing the incoming XML data element based on delimiters to determine the source web page, to include a means of converting the template after combining the plurality of parts for the web page with the template to form the web page into HTML so as to be displayed by the browser as taught by Uhler. One of the ordinary skill in the art would have been motivated to modify this combination, because Chen and Uhler are from the same field of endeavor of providing an architecture for creating extensible and scalable web applications, and provides a server object handler uses a set of HTML/XML templates to process content and the final filter performing the XML to HTML conversion, which is consistent with the ultimate consumer of the content, and deliver to the requestor (Uhler col. 14, line 60 through col. 15, line 10).

Regarding claims 14 and 17, Chen teaches **incoming XML data element is a portion of a web page in which that data to be displayed is changing and said token is an existing web page.** Specifically, Chen discloses a first parser for receiving a message from a browser (Chen col. 1, lines 35-50). Also, Chen discloses a standard XML parser 305 takes the input XML 125 and DTD 115, and generates an intermediate structure, a tree 355 or an array 355', which serves as part of the input data to a merge algorithm 335. The XML parser 305 may be a client side application, which may serialize tree elements into an array of hypertext markup language (HTML) components 355', or a server side stand-alone application, which may construct the tree structure 355 (See FIGS. 10A and 10B). After parsing the return document DTD 135, the DTD parser 315 creates a template 365 in either array format 605 or tree structure 615, as shown in FIGS. 10A and 10B, respectively (Chen col. 6, lines 5-20 also Fig. 7).

Regarding claims 15 and 18, Chen teaches **wherein said modified XML data element is the web page to be displayed.** Specifically, Chen discloses after parsing the return document DTD 135, the DTD parser 315 creates a template 365 in either array format 605 or tree structure 615, as shown in FIGS. 10A and 10B, respectively (Chen col. 6, lines 5-20 also Fig. 7). Also, Chen discloses a merging algorithm, which is implemented to merge the message with the return template for providing a return message to the browser having portions of the return template with data entered therein. (Chen Col. 1, lines 45-50).

Conclusion

6). Any inquiry concerning this communication or earlier communications from the examiner should be directed to Quoc A. Tran whose telephone number is 571-272-8664. The examiner can normally be reached on 9AM - 5PM EST.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Herndon R. Heather can be reached on 571-272-4136. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

H Herndon
Heather R. Herndon
Supervisory Patent Examiner
Technology Center 2100

Quoc A. Tran
February 28, 2007